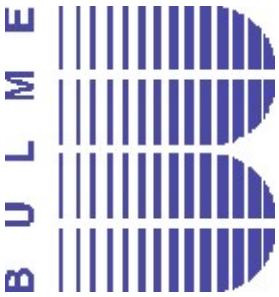
HTBLuVA Graz – Gösting

Ibererstraße 15-21

8051 Graz



Abteilung für Elektronik und technische Informatik

UwU-Bot

Diplomarbeit im Fach Technische Informatik

Eingereicht von: Lukas-Gerald Greiss
Manuel Greiler
Nico Huber

im Schuljahr 2022/2023

der Klasse 5CHEL

bei DI Franz Albert Patz

am 31. März 2023

Inhaltsverzeichnis

Seite

Abbildungsverzeichnis

Seite

Tabellenverzeichnis

Seite

Quellenverzeichnis

Bibliotheken:

ESP32 Display Bibliothek: https://github.com/mrfaptastic/ESP32-HUB75-MatrixPanel-DMA

ESP32 Schrittmotor Bibliothek: https://www.airspayce.com/mikem/arduino/AccelStepper/

Android Bluetooth Bibliothek: https://github.com/harry1453/android-bluetooth-serial

Links für Bilder und Infos:

https://techexplorations.com/guides/esp32/begin/power/

https://howtomechatronics.com/tutorials/arduino/stepper-motors-and-arduino-the-ultimate-guide/

https://miliohm.com/how-to-drive-a-stepper-motor-easily-using-a4988-and-arduino/

https://randomnerdtutorials.com/esp32-dual-core-arduino-ide/

https://randomnerdtutorials.com/esp32-pinout-reference-gpios/

https://stackoverflow.com

Weitere Links:

Bitmap Converter: https://javl.github.io/image2cpp/

GitHub des UwU-Bots: https://github.com/Manyullyn17/UwU-Bot

Datenblattverzeichnis

ESP32: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

Schaltregler: https://docs.rs-online.com/402c/0900766b816eda0f.pdf

Schrittmotor-Treiber: https://components101.com/sites/default/files/component_datasheet/A4988%20Stepper%20Motor%20Driver%20Module.pdf

Schrittmotor: https://www.omc-stepperonline.com/index.php?route=product/product/get_file&file=1340/17HM15-1004S_Full_Datasheet.pdf

Akku: https://de.aliexpress.com/item/32880431474.html

DC Buchse: https://docs.rs-online.com/edae/0900766b81545613.pdf

1 Ziele und Inhalte

1.1 Kurzbeschreibung

Gebaut werden soll ein kleiner, akkubetriebener Roboter aus Holz. Dieser fährt auf zwei Rädern, die mit Schrittmotoren betrieben werden. Die Fahrbewegung kann via Android App ferngesteuert werden. Zusätzlich besitzt der Roboter eine LED RGB Matrix als Display, worauf er Emotionen zeigen kann. Auch die Emotionen können über ein Menü in der App ausgewählt werden.

1.2 Abstract

The final product is a small, battery-powered robot that is made of wood and runs on two wheels that are driven by stepper motors. Its movement can be controlled via an Android App that connects via Bluetooth. In addition, the robot has an LED RGB matrix as a display, on which it can show emotions. The emotions can also be selected via a menu in the app.

1.3 Untersuchungsanliegen der individuellen Themenstellungen

Lukas-Gerald Greiss:
Ansteuerung der LED RGB Matrix und das Empfangen der Daten über den Mikrocontroller (ESP32) mit Arduino IDE. Weiters sinnvolle Auswahl der Hardwarekomponenten (Motoren, LED Matrix, µC)

Nico Leon Huber:
Android-App Programmierung über Android Studio und Senden der Daten an den Mikrocontroller (ESP32). Außerdem kümmert er sich um den Aufbau des Gehäuses. Verantwortlich für die Anschaffung der Bauteile.

Manuel Greiler:
Konstruktion bzw. Gehäusedesign (3D-Modell) und die Ansteuerung der Motoren über den Mikrocontroller (ESP32). Recherche von Softwarekomponenten (App und µC). Weiters verantwortlich für die Fehlersuche und Koordinationsaufgaben innerhalb des Teams. Aufgrund der Expertise Support bei Programmierung und Hardware-Design.

Geplantes Ergebnis der Prüfungskandidaten

- Fahrwerk aus Schrittmotoren, deren Drehzahl über die App eingestellt werden kann
- Android-App zur Ansteuerung des ESP32 über Bluetooth und Kontrolle des Fahrwerks und des Displays
- 3-D Modell und Bau des Gehäuses

1.4 Meilensteine

- 24.11.2022 Fahrwerk funktioniert
- 15.12.2022 Display Ansteuerung
- 12.01.2023 App funktioniert ohne Kommunikation
- 02.02.2023 Kommunikation ESP32 und App
- 09.02.2023 Gehäusebau
- 16.02.2023 Verkabelung und Zusammensetzen der Teilkomponenten
- 23.02.2023 Fehlerbehebung
- 28.02.2023 Fertigstellung der Diplomarbeit

1.5 Ausgangslage

Das Endprodukt soll ein kleiner Roboter sein, der über eine Smartphone - App gelenkt werden und über ein Menü diverse Emotionen ("Face-Expressions") auf der LED RGB Matrix anzeigen kann.

1.6 Wozu ist der Roboter gut

Der Roboter soll nur ein Spielzeug darstellen. Da wir die Idee süß fanden, wollten wir einen Roboter erschaffen. Er hat die Fähigkeit zu fahren und Gesichter verschiedener Emotionen darzustellen. Er ist der perfekte Roboter für eine schöne Zeit zu Hause. Wenn man will, kann man damit im Haus herum, aber er macht auch nur so im Zimmer stehend einen guten Eindruck. Außerdem ist die Gesichter-Vielfalt auch ein guter Weg seine Emotionen auszudrücken, wenn man Probleme hat diese zu äußern. Man stelle sich vor, ein Kind kommt in die Pubertätsphase, gerade der Zeitpunkt wo das Ausdrücken der Emotionen ein großes Problem ist, dann wäre der UwU-Bot ein gutes Mittel diese Emotionen zu übermitteln.

1.7 Anwender

Sollte dieser Bot es auf den Markt schaffen, was unwahrscheinlich ist, da er keine neuen oder innovativen Features hat, wäre dieser für jede Altersklasse geeignet. Von der Grundidee und mit etwas mehr Arbeit könnte dieser Roboter ein gutes Spielzeug sein, dass am Markt eine Berechtigung hätte.

1.8 Randbedingungen

Es gab bei dem Projekt keine Unterstützung von Firmen, es war nur eine kleine Idee, die dann zur Realität wurde. Das Projekt wird von Lukas Greiss, Nico Huber und Manuel Greiler finanziert und realisiert.

2 Projektplanung

2.1 Über uns

2.1.1 Lukas-Gerald Greiss

Mein Name ist Lukas-Gerald Greiss ich bin in Graz geboren und im Burgenland aufgewachsen. Ich unternehme gerne etwas mit Freunden, aber wenn ich mal allein bin, dann spiele ich Computerspiele oder mache Musik. Da ich für Informatik und Elektronik äußerst viel Interesse zeige, entschloss ich mich die HTL Bulme zu besuchen, um meinem Wunsch nachzugehen, in Zukunft einen Beruf in diesem Bereich auszuüben.

2.1.2 Nico Leon Huber

Mit rumänischen Wurzeln, in Österreich geboren, erkämpfte ich mir die Akzeptanz jener Schülerinnen und Schüler, die mit mir die Grundschule gemeinsam besuchten. Dabei versteckte ich mich meist hinter den Monitoren meines alten PCs zuhause, um mich an den Freuden des „Gamens“ zu laben. Seitdem hatte ich den Traum in meiner fernen Zukunft mit Computern zu hantieren und besuchte daher die HTL Bulme. Hier entdeckte ich vielerlei interessante Themen und Gebiete, an denen ich meine Zukunft orientieren möchte, aber leider auch vieles, das meinem Interesse nicht folgen konnte. Meine Hobbys sind hierbei abgesehen vom „Gamen“, mich ab und zu unter die Menge zu mischen oder diversen Aktivitäten wie dem Reisen nachzukommen.

2.1.3 Manuel Greiler

Geboren in Kärnten, bin ich 2017 in die Steiermark gezogen und habe im darauffolgenden Jahr meine Karriere an der HTL Bulme begonnen. Schon als kleines Kind war ich immer interessiert an Computern und Technik, weshalb die Bulme dann der offensichtliche nächste Schritt war. Meine Hobbies umfassen Gaming, Anime, Manga und Zeit mit Freunden verbringen. Des Weiteren beschäftige ich mich gerne mit allem, das mit Computern und Technik zu tun hat.

2.2 Pflichtenheft

Der Roboter soll eine akkubetriebene würfelförmige Box aus Holz mit zwei Hinterrädern als Antrieb und Möbelrollen als Vorderräder sein. Zur Lenkung wird eine Panzersteuerung verwendet. Die Räder sollen mit Schrittmotoren angetrieben werden, welche von einem ESP32 kontrolliert werden. Außerdem soll er eine LED Matrix als Display besitzen, auf welcher diverse Emotionen angezeigt werden können, welche ebenfalls mit dem ESP23 angesteuert wird. Dieser soll via einer Android App über eine Bluetooth-Verbindung Signale bekommen, um die Motoren und das Display fernzusteuern. Die App soll zwei Joysticks besitzen, welche für die Steuerung der Motoren verantwortlich sind und ein Dropdown-Menü um die Emotionen auszuwählen.

2.3 Aufgabenteilung

2.3.1 Lukas-Gerald Greiss

Ich habe das Programmieren der Ansteuerung von der LED-Matrix und die Leiterplatte übernommen. An der LED-Matrix zu arbeiten, umfasst es, die Bilder richtig auf LED-Matrix zu übertragen. Die Leiterplatte habe ich mittels Eagle verwirklicht, es ist eine Doppelseitige Platine, sie bildet das Herzstück des UwU-Bots.

2.3.2 Nico Leon Huber

Ich habe mich vollständig dem Design und dem Gehäuse gewidmet und anhand des 3D-Modells, das ich erstellt hatte, hatte ich einen guten Überblick über die Maße und die Zusammensetzung der einzelnen Bauteile. Außerdem war ich für die Beschaffung der einzelnen Bauteile und Materialien verantwortlich. Weiteres hatte ich mir als Challenge gesetzt, die App zu programmieren. Dabei habe ich das Menü erstellt und programmiert, welches für die Auswahl der Emotionen auf der LED-Matrix verantwortlich ist. Aufgrund von Komplikationen mit den Joysticks übernahm Manuel Greiler die Programmierung dieser.

2.3.3 Manuel Greiler

Ich habe mich um das Design der Räder, die Ansteuerung der Motoren, die Übertragung der Daten von App zu ESP32 und später auch die Programmierung der restlichen App gekümmert. Die Räder habe ich mittels Fusion360 modelliert und dann 3D gedruckt. Die Steuerung der Motoren und das Empfangen der Daten am ESP32 habe ich mit der Arduino IDE programmiert. Des Weiteren habe ich die Programmierung der Joysticks und der Bluetooth Übertragung der App übernommen, da Nico Huber Probleme hatte diese zu vervollständigen.

3 Projektrealisierung

3.1 Blockdiagramm

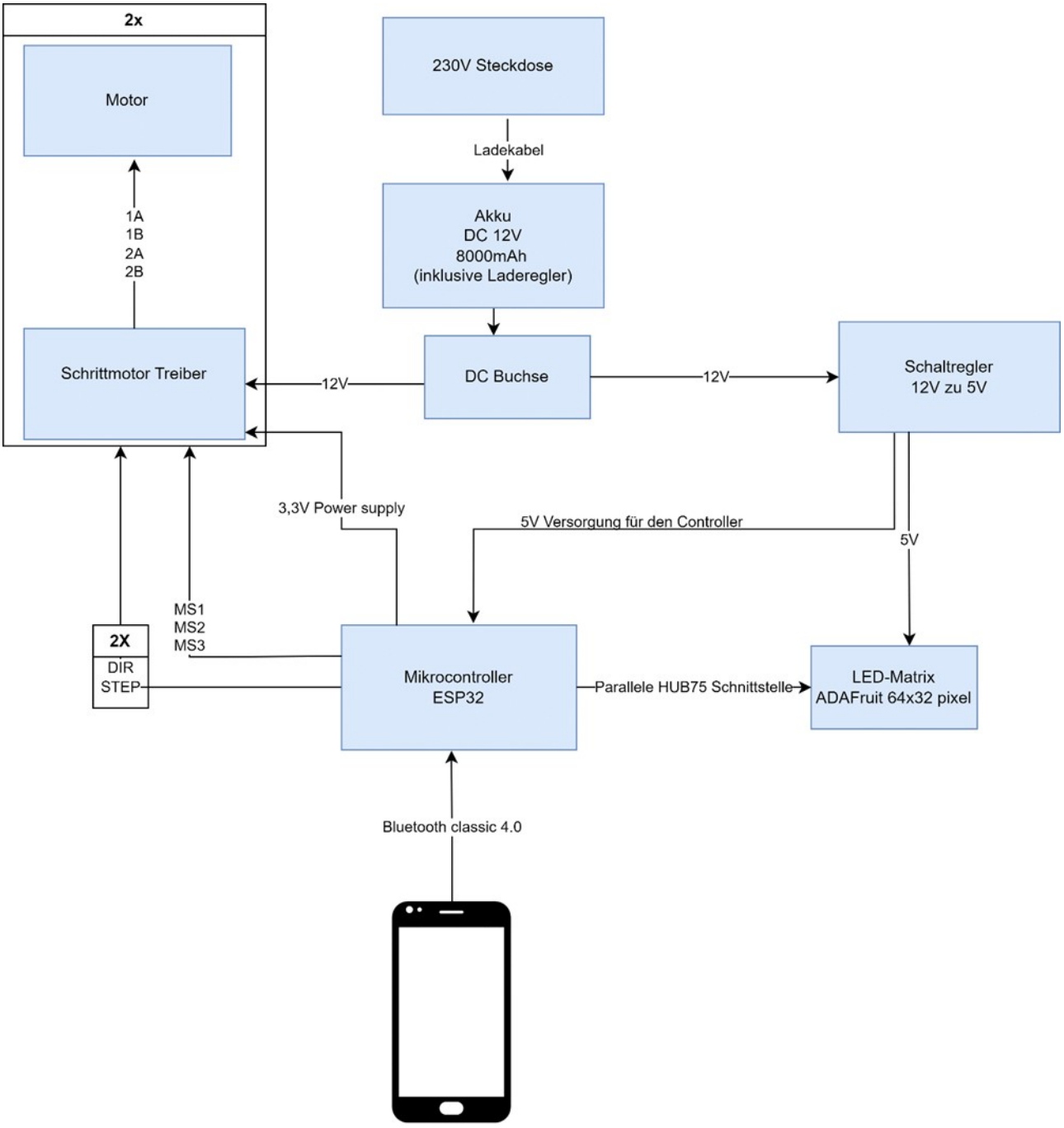


Abbildung 1 - Blockdiagramm

Der Akku wird mit dem mitgelieferten Netzteil aufgeladen. Der Akku versorgt den ganzen Roboter mit Strom. Der Akku gibt 12V, erstens zu dem Schrittmotor-Treiber und zweitens mittels einer Buchse zu einem Schaltregler, der die 12V zu 5V schaltet. Von dort aus gehen die 5V zu dem Mikrocontroller, der Mikrocontroller steuert die LED-Matrix über eine parallele HUB75 Schnittstelle an, als auch die Motoren über den Schrittmotor-Treiber, 5V braucht auch die 64x32 Pixel LED-Matrix. Der Mikrocontroller hat einen eingebauten Spannungsregler mit dem er die 5V nochmal auf 3,3V runterregelt, diese werden für die Logik der Schrittmotoren gebraucht.

Der Mikrocontroller gibt die Informationen an die MS, DIR und STEP Eingänge des Schrittmotor-Treibers weiter, die MS Pins sind für das Microstepping, Microstepping ist eine Funktion der Schrittmotoren, mit der der Motor nicht nur ganze Drehungen durchführen kann, sondern auch sogenannte Micro Schritte durchführen kann. Die genauere Funktion des Microsteppings wird in Kapitel 3.4.2.2 genauer erklärt. Die DIR Schnittstelle bestimmt, in welche Richtung sich er Motor drehen soll und die STEP Schnittstelle macht mit jedem Puls einen Schritt, was dazu führt, dass sich der Motor dreht.

Die 1A, 1B, 2A und 2B Leitung sind schlichtweg Stromleitungen jeweils 2-mal Plus und 2-mal Minus, mit denen beide Spulen des Motors gesteuert werden. Der Microcontroller wird mittels Bluetooth 4.0 angesteuert, mit dem Handy kann man die LED-Matrix und die Motoren ansteuern.

3.2 UwU-Bot App

Die App besteht aus grundsätzlich zwei notwendigen Komponenten. Einem Menü, mit welchem man die verschiedenen Gesichter auswählen kann, die dann schlussendlich bei der LED-Matrix angezeigt werden können und zwei Joysticks, die für das Fahrwerk zuständig sind. Außerdem kommuniziert die App mit dem ESP32 über eine Bluetooth Classic 4.0 – Verbindung, indem diese die nötigen Daten (die einzelnen Gesichtsausdrücke oder die Standpunkte der Joysticks) überträgt.

3.2.1 Design der App

Im Voraus wurde erstmal eine Skizze angefertigt, wo was sein sollte und wie genau alles ausschauen sollte. Hierbei wurde dafür gesorgt, dass man auch genug Platz für alles hat.

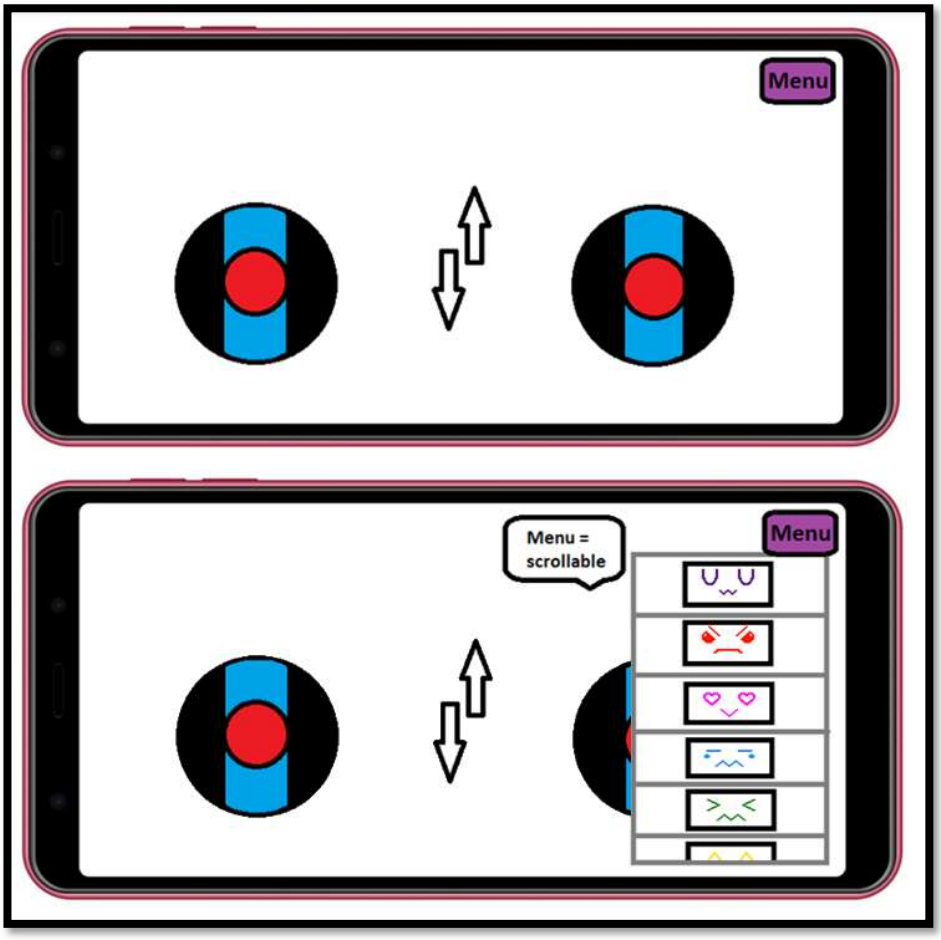


Abbildung 2 - Skizze der App

Wie man sieht, sollen die Joysticks nur auf und ab gehen und im Menü kann man scrollen.

3.2.2 Menü

Das Menü besteht aus einem Button, das Bilder aufruft, wodurch man bestimmen kann, welche Gesichter auf der LED-Matrix angezeigt werde. Da es zu viele Gesichter gibt für ein simples Menü, muss man durch das Menü scrollen können. Eine vollständige Liste der Gesichter kann in Kapitel 3.5 gefunden werden.

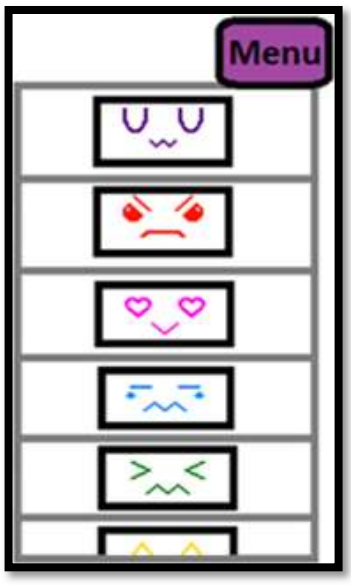


Abbildung 3 - Menü

3.2.3 Joysticks

3.2.3.1 Anforderungen

Die App soll zwei Joysticks haben, die verwendet werden, um die Motoren des Roboters zu steuern. Die Joysticks sollen nur nach oben und unten bewegt werden und Werte von -100 bis 100 liefern, welche die Geschwindigkeit der Räder in Prozent darstellen, wobei bei negativen Werten die Räder sich rückwärts drehen. Außerdem sollen sie, wenn sie losgelassen werden, automatisch zum Nullpunkt zurückkehren.

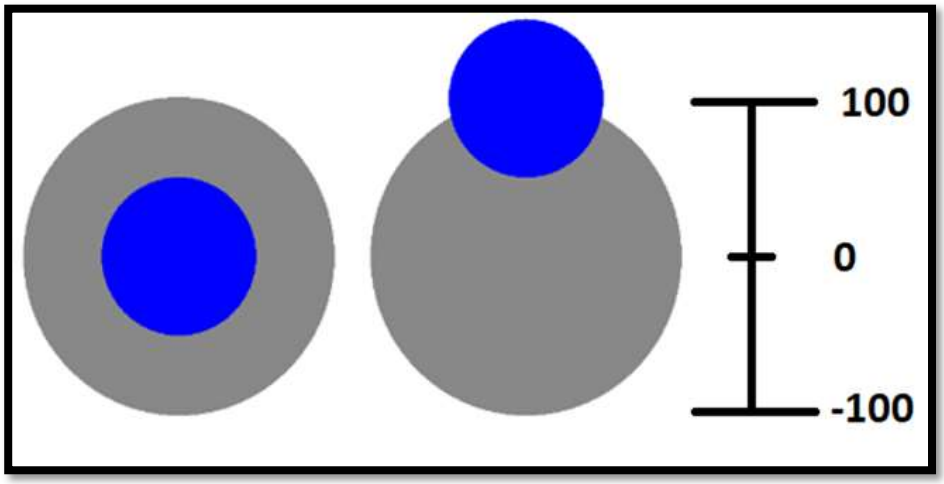


Abbildung 4 - Joysticks

3.2.3.2 Implementierung

Die Joysticks sind komplett selbst implementiert, da keine vorgefertigte Joystick-Bibliothek gefunden werden und auch zum Funktionieren gebracht werden konnte. Diese Implementation ist „purpose-built“, das heißt, dass die Joysticks genau für unseren Anwendungszweck angepasst sind und nicht ohne den Quellcode zu verändern für andere Anwendungszwecke verwendbar sind.

3.2.3.3 Funktionalität

Wenn die Joysticks bewegt werden, wird automatisch eine Funktion aufgerufen, die verwendet wird, um die Werte an den ESP32 zu senden. Die Werte werden bereits intern so verarbeitet, dass sie in dem gewünschten Wertebereich von -100 bis 100 liegen und auf die nächste Fünf gerundet sind, da eine genauere Steuerung der Drehgeschwindigkeit nicht notwendig ist und dadurch die aktualisierten Werte seltener gesendet werden müssen.

Die Joysticks können einfach in der Größe verändert werden, ohne dass der Wertebereich sich verändert und es können so viele Joysticks auf einmal verwendet werden wie gewünscht.

3.3 Datenübertragung und Verarbeitung

3.3.1 Bluetooth Übertragung

3.3.1.1 Anforderungen

Die Kommunikation zwischen der App und dem ESP32, der den Roboter steuert, soll via Bluetooth 4.0 realisiert werden. Es sollen die Position der Joysticks, die für die Kontrolle der Motoren zuständig sind, und der gewünschte Gesichtsausdruck, wenn einer im Menü der App ausgewählt wird, in Echtzeit übermittelt werden.

Der ESP32 muss keine Daten an die App zurückschicken und die App muss auch keine Daten empfangen können, da eine Einweg-Kommunikation ausreichend ist.

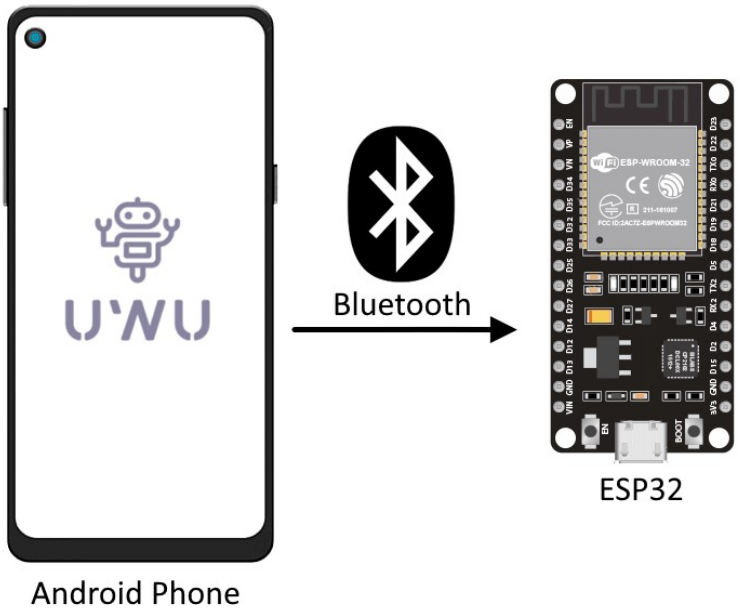


Abbildung 5 - Übertragung App - ESP32

3.3.1.2 Senden der Daten

Die App verbindet sich mit dem ESP32 und sendet bei einer erfolgreichen Verbindung die Positionen der beiden Joysticks und den gewählten Gesichtsausdruck. Dafür wird die Android-Bluetooth-Serial Bibliothek verwendet.

Die Bibliothek erlaubt es, eine Liste mit allen verfügbaren Bluetooth Geräten abzurufen, sich mit einem Gerät seiner Wahl zu verbinden und Daten zu senden und empfangen. Die App verbindet sich automatisch mit dem Gerät, welches den zuvor festgelegten Namen des ESP32 besitzt, und ist, wenn die Verbindung erfolgreich ist, bereit Daten zu senden.

Daten werden gesendet, wenn einer oder beide Joysticks bewegt werden oder im Menü ein Gesichtsausdruck ausgewählt wird. Die Bluetooth-Verbindung wird automatisch beendet, sobald die App geschlossen wird.

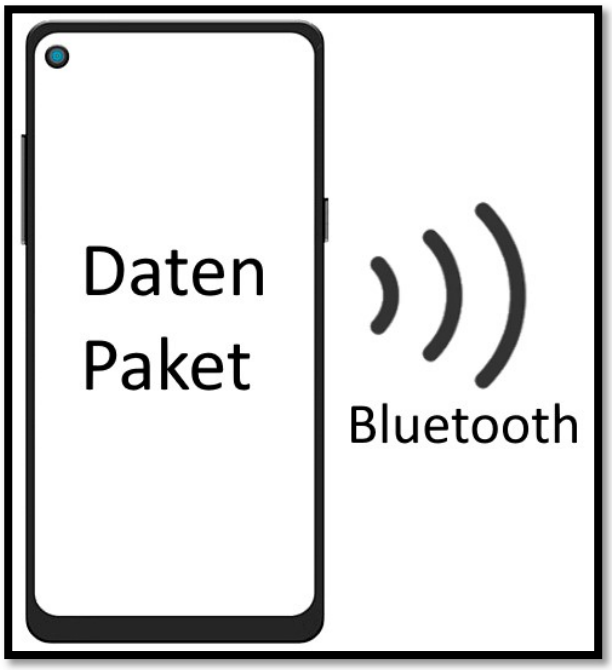


Abbildung 6 - Senden der Daten

3.3.1.3 Multithreading

Um sicherzustellen, dass die Daten so zeitnah wie möglich gesendet werden, wird ein eigener Thread verwendet. Hierfür wird die eingebaute „Thread“ Klasse verwendet.

Der Thread wird gestartet, sobald die App sich mit dem ESP32 verbunden hat. Dieser bekommt die Daten in Form eines Strings und verwendet die „sendMessage()“ Funktion, um diesen String zu senden.

Falls keine Bluetooth-Verbindung besteht, wird nicht versucht, Daten zu senden, da sonst eine Exception auftreten könnte, welche die App abstürzen lässt. Da die App nicht versucht Daten zu senden, wenn sie nicht mit dem ESP32 verbunden ist, kann dies nicht passieren.

3.3.1.4 Empfang der Daten

Der ESP32 empfängt die Daten, die von der App gesendet werden, und verarbeitet diese. Dies wird mit Hilfe der eingebauten BluetoothSerial Bibliothek realisiert. Diese Bibliothek erlaubt es dem ESP32 eine Bluetooth-Verbindung herzustellen und Daten zu senden und zu empfangen.

Zu Beginn wird in der Setup-Funktion die „begin()“ Funktion aufgerufen, in welcher der Anzeigename des ESP32 festlegt wird. Dies geschieht sobald der ESP32 mit Strom versorgt wird und sorgt dafür, dass eine Verbindung mit der App hergestellt werden kann.

Mit der Funktion „readStringUntil()“ kann, solange Daten zum Einlesen verfügbar sind, ein String bis zu einem bestimmten Zeichen (den Terminator) eingelesen werden. Dadurch können die einzelnen Instruktionen eingelesen und dann verarbeitet werden, ohne dass der eingelesene String in mehrere Segmente geteilt werden muss.

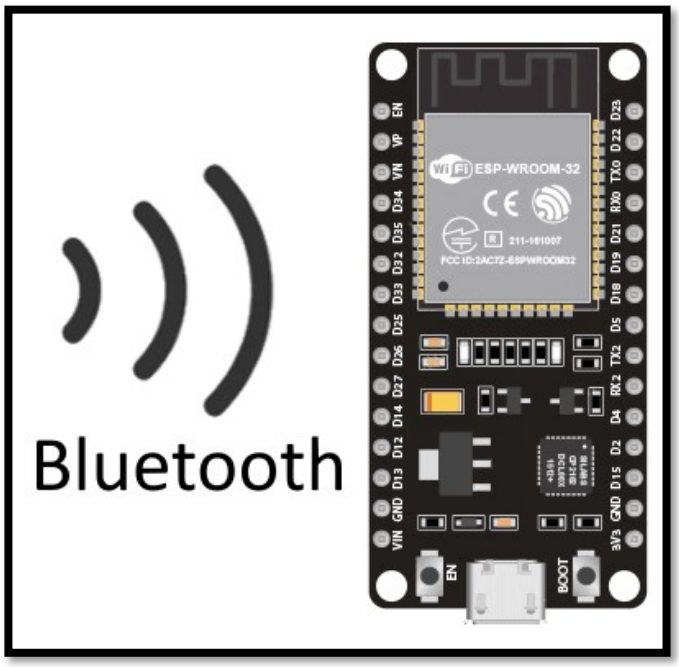


Abbildung 7 - Empfang der Daten

3.3.2 Verarbeitung der Daten

Die Daten werden als einzelne Pakete gesendet, die aus einem Befehl und einem Wert bestehen. Die Pakete werden als ein einzelne Strings ohne Leerzeichen gesendet und jeweils mit einem Null-Terminator am Ende getrennt.

Der Befehl ist ein einzelner Großbuchstabe und gibt an, wie der darauffolgende Wert zu interpretieren ist. Der Wert ist immer eine Ganzzahl und wird entsprechend des davorstehenden Befehls interpretiert. Nachdem der Befehl und der Wert eingelesen wurden, wird dieser ausgeführt.

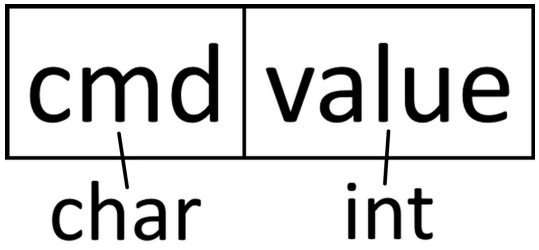


Abbildung 8 - Diagramm Datenpaket

Die Befehle „R“ und „L“ setzen die Geschwindigkeit, mit der sich die Motoren drehen sollen, wobei „R“ den rechten Motor und „L“ den linken Motor ansteuert. Der darauffolgende Wert reicht von -100 bis 100 und stellt die Geschwindigkeit des Motors in Prozent dar, wobei eine negative Zahl bedeutet, dass der Motor sich rückwärts drehen soll.

Der Befehl „F“ steht für „Face“ und setzt den Gesichtsausdruck, der am Display angezeigt werden soll. Der darauffolgende Wert reicht von 1 bis 8 und besagt, welches Gesicht angezeigt werden soll.

3.4 Fahrwerk

3.4.1 Schrittmotoren

Der Roboter fährt mit Hilfe von zwei Schrittmotoren, an deren Achsen 3D gedruckte Räder befestigt sind. Die Motoren sind Bipolar Schrittmotoren vom Typ Nema17, mit einem Haltemoment von 26 Ncm, 1.8º Schrittweite und einer maximalen Geschwindigkeit von 200 U/min. Die Motoren benötigen eine Spannungsversorgung von 12V und haben eine Stromaufnahme von bis zu 0.4A.

Diese Motoren sind besonders gut geeignet für das Fahrwerk des UwU-Bots, da sie stark genug sind, um diesen ohne große Probleme fortzubewegen und nur eine Spannung von 12V benötigen, welche relativ einfach mit einem wiederaufladbaren Akku zur Verfügung gestellt werden kann.

Des Weiteren sind sie sehr leicht durch die Verwendung eines Schrittmotor-Treibers anzusteuern, wodurch keine externen Transistoren zur Steuerung der höheren Spannung verwendet werden müssen und nur zwei Pins pro Motor für die Ansteuerung benötigt werden.

Abbildung 9 - Nema17 Schrittmotor

3.4.2 Schrittmotor-Treiber

Die Motoren werden mit jeweils einem WJMY A4988 Schrittmotor-Treiber gesteuert. Die Treiber können verwendet werden, um jeweils einen Motor mit vier Anschlüssen zu steuern.

Sie unterstützen Microstepping für bessere Schrittgenauigkeit und werden mit nur zwei Pins gesteuert. Der Treiber besitzt auch ein Potentiometer, mit dem der Strom, der durch den Motor fließt, limitiert werden kann, wenn man diesen mit einer höheren Spannung als der Nennspannung betreibt, um höhere Schrittraten zu erzielen.

Diese Funktion ist wichtig, um mögliche Schäden an den Motoren aufgrund zu hoher Ströme zu verhindern.

Die folgenden Verbindungen zwischen Schrittmotor und Schrittmotor-Treiber sind speziell für die von uns verwendeten Schrittmotoren und können bei Verwendung von anderen Motoren abweichen. In diesem Fall sind die Verbindungen aus dem dazugehörigen Datenblatt zu entnehmen.

Tabelle 1 - Treiber zu Motor

Pin	Kabel
1A	Schwarz
1B	Blau
2A	Grün
2B	Rot

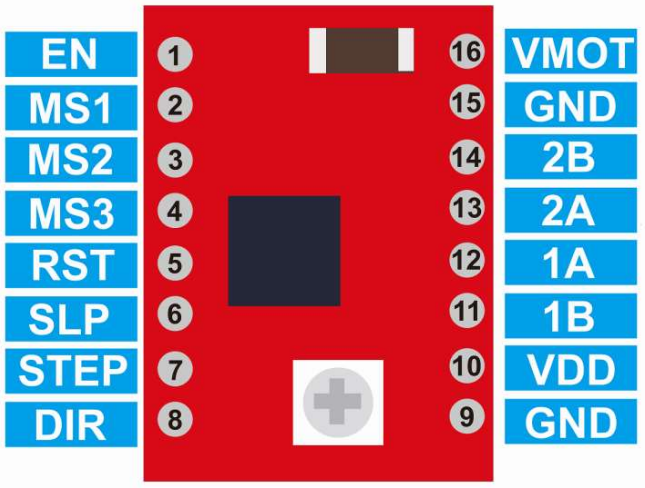


Abbildung 10 - Schrittmotor-Treiber Pinout

3.4.2.1 Anschließen an ESP32 und Motoren

Die Motoren können wie folgt an den ESP32 angeschlossen werden:

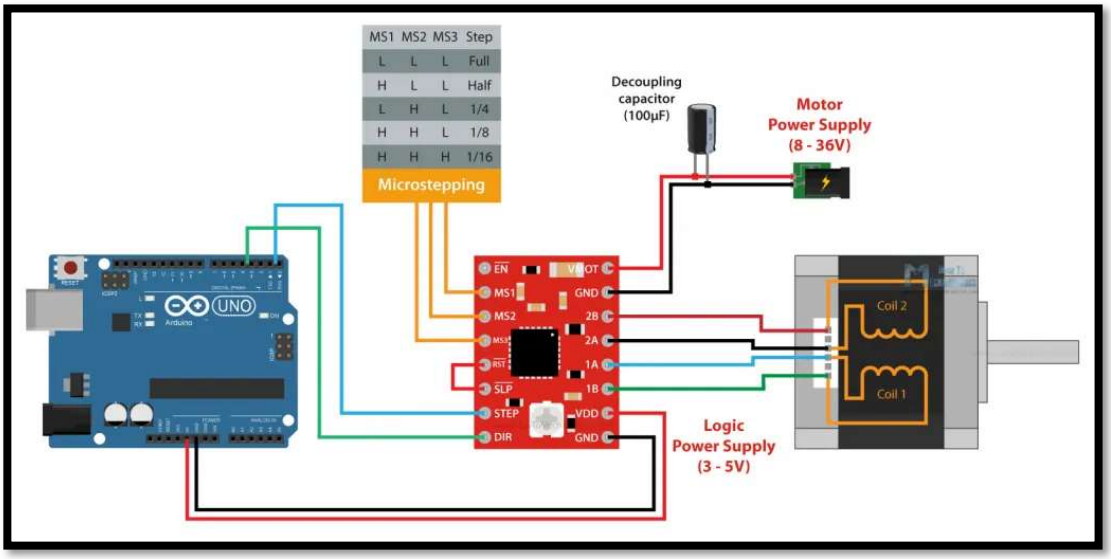


Abbildung 11 - Schrittmotor Schaltung

Die hier dargestellten Verbindungen dienen nur zur Erklärung und repräsentieren nicht die tatsächlich verwendeten Verbindungen. Die genauen hier verwendeten Verbindungen können im Schaltplan in Kapitel 3.6 gefunden werden.

Der STEP Pin lässt den Motor bei einem Puls um einen Schritt weiter rotieren und der DIR Pin bestimmt die Drehrichtung des Motors. Die Pins MS1, MS2 und MS3 werden für Microstepping verwendet und die Pins 1A, 1B, 2A und 2B werden mit dem Motor verbunden. Wie genau diese Pins mit dem Motor verbunden werden müssen, ist dem jeweiligen Datenblatt des Motors zu entnehmen.

Der Treiber muss mit 3-5V als Logikspannung und 8-36V als Spannungsversorgung für den Motor versorgt werden. Die RST und SLP Pins können, wenn die Reset und Sleep Funktionen des Treibers nicht verwendet werden, verbunden werden, um das Board in den aktiven Zustand zu versetzen.

Der EN Pin steht für Enable und ist active-low. Das bedeutet, dass das Board aktiv ist, wenn dieser auf LOW gesetzt wird. Dieser Pin muss nicht verbunden werden und ist auch ohne Verbindung aktiv.

3.4.2.2 Microstepping

Die Treiber unterstützen fünf Microstepping Modi: Voll, 1/2, 1/4, 1/8 und 1/16. Microstepping erlaubt es kleinere Schrittweiten zu erzielen, als der Motor normalerweise unterstützt. Dies wird durch die Verwendung von PWM (Pulsweitenmodulation) realisiert, wodurch der Motor zwischen zwei Schritten gehalten werden kann.

Die Verwendung von Microstepping erlaubt nicht nur eine höhere Auflösung, sondern reduziert auch Vibrationen und sorgt für eine flüssigere Bewegung und ein konstanteres Drehmoment. Microstepping hat jedoch auch Nachteile, wie zum Beispiel verringertes Drehmoment und Haltemoment.

Die Pins MS1, MS2 und MS3, welche zur Auswahl des Microstepping Modus verwendet werden, sind mit drei IO Pins des ESP32 verbunden, damit diese per Software kontrolliert werden können, anstatt fest verdrahtet zu sein oder mit Jumpern auf der Leiterplatte kontrolliert zu werden.

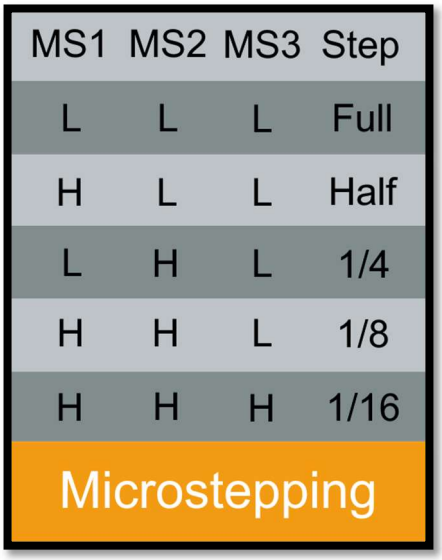


Abbildung 12 - Microstepping

3.4.3 Ansteuerung mit Arduino Bibliothek

Zur Ansteuerung der Schrittmotor-Treiber verwenden wir die AccelStepper Arduino Bibliothek. Diese unterstützt 2, 3 und 4 Pin Schrittmotoren und Treiber und beinhaltet Funktionen zum Setzen der gewünschten Drehgeschwindigkeit, der Beschleunigung, mit der der Motor auf die gewünschte Drehgeschwindigkeit beschleunigt und einer bestimmten Anzahl an Grad, um welche sich der Motor drehen soll.

Damit der ESP32 den Schrittmotor-Treibern ein Step-Signal sendet, muss so oft wie möglich die Funktion „runSpeed()“ aufgerufen werden, da sonst die Treiber nicht schnell genug ein Signal bekommen, um den Motor weiterzudrehen.

Dies kann durch die Verwendung von Tasks gelöst werden. Da der ESP32 zwei Kerne hat, kann ein dedizierter Task für diese Funktion auf einem Kern laufen, während der Rest des Codes auf dem anderen Kern läuft.

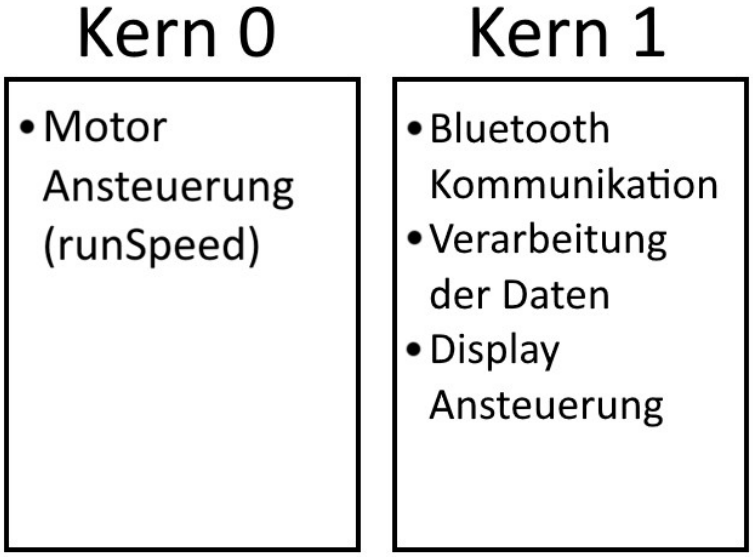


Abbildung 13 - ESP32 Multitasking

3.4.4 Räder

Die Räder wurden in Fusion 360 modelliert und daraufhin mit einem 3D-Drucker gedruckt. Sie wurden so gestaltet, dass sie auf den meisten Oberflächen gut haften und der Roboter fahren kann, ohne die Räder durchzudrehen. Die einzige Schwachstelle der Räder sind sehr glatte Oberflächen, da sie aus nicht gummiertem PLA Kunststoff bestehen.

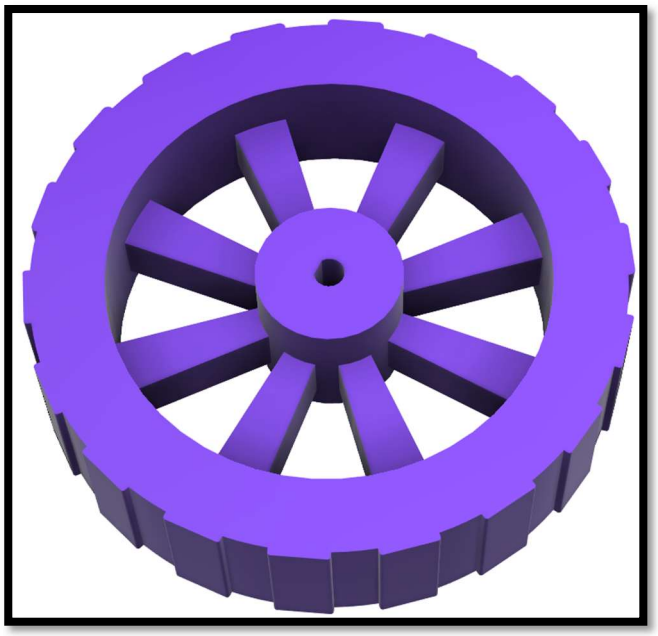


Abbildung 14 - 3D-Modell Rad

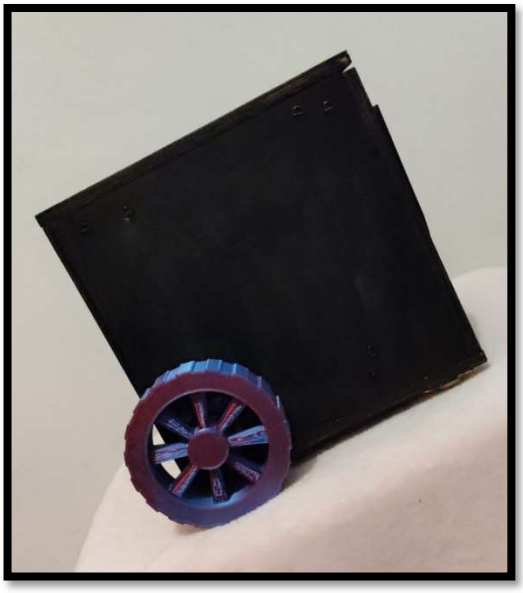


Abbildung 15 - Bot auf schräger Oberfläche

3.4.5 Lenken

Da der Roboter nur zwei angetriebene Hinterräder hat und keine Vorderräder, die mit einem Lenksystem versehen sind, wird er mit sogenannter „Panzersteuerung“ gelenkt. Das heißt, dass die beiden Räder unabhängig voneinander nach vorne bzw. hinten fahren und so den Roboter drehen können.

Um dieses Steuerungsschema zu erleichtern, kommen Möbelrollen anstatt konventionellen Vorderrädern zum Einsatz. Diese Rollen können sich in jede Richtung drehen und ermöglichen es dem Roboter, sich ohne viel Widerstand komplett um die Hinterachse zu drehen, was einen engeren Wendekreis ermöglicht.

Diese Lösung reduziert die Komplexität des Lenksystems erheblich, da keine weiteren Motoren oder ähnliches angesteuert werden müssen, und es mit nur zwei Motoren möglich ist, den Roboter zu lenken.

Der Roboter kann dank dieses Steuerungsschemas vorwärts und rückwärts fahren, während des Fahrens lenken und, wenn die beiden Räder sich in entgegengesetzte Richtungen drehen, sogar auf der Stelle drehen. Die Wendegeschwindigkeit kann durch das Verhältnis der Geschwindigkeit der beiden Räder genau gesteuert werden. Dies bedeutet, dass der Roboter, wenn das linke Rad sich schneller dreht als das rechte, eine Rechtskurve macht. Die folgende Abbildung illustriert, wie er sich auf der Stelle dreht.

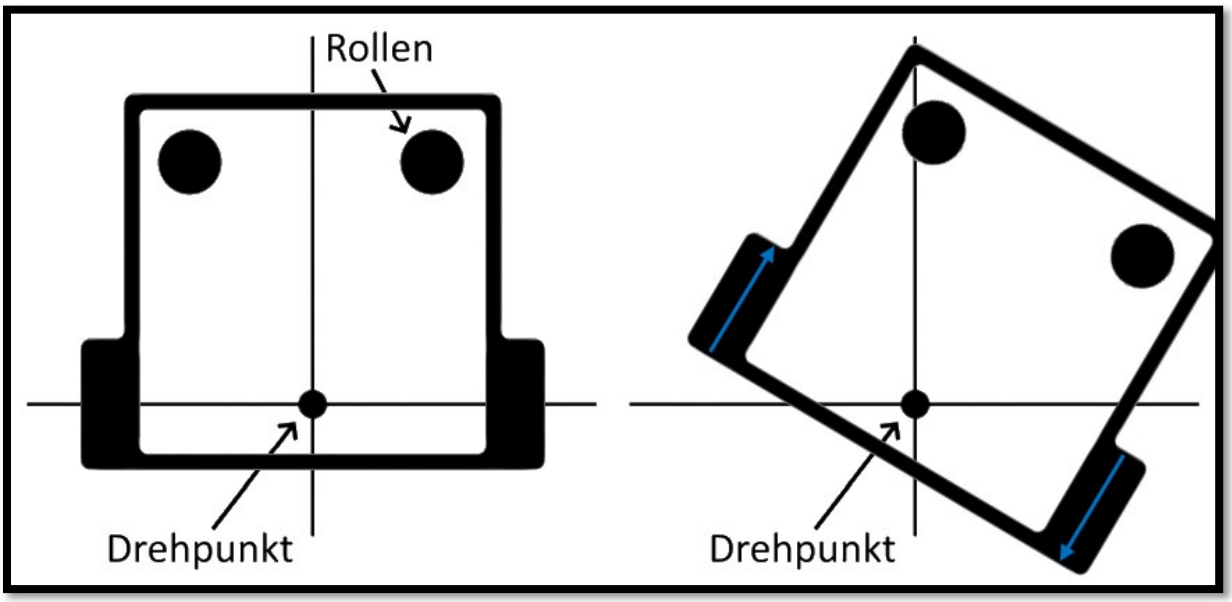


Abbildung 16 - Lenkung mit Panzersteuerung

3.5 LED-Matrix

Die LED-Matrix wird via ESP32 angesteuert, es wird auch eine Bibliothek (ESP32-HUB75-MatrixPanel-DMA) verwendet. Diese Bibliothek vereinfacht schon einiges, zum Beispiel kann man mit dieser: Linien, Buchstaben, Kreise und für unser Projekt wichtig Bitmaps auf die LED-Matrix kompilieren. Nach einiger Recherche wurde ein Bitmap Converter gefunden mit diesem Converter können einfache PNGs zu einer Bitmap umgewandelt werden, welche durch die Bibliothek auf dem Display angezeigt werden können.

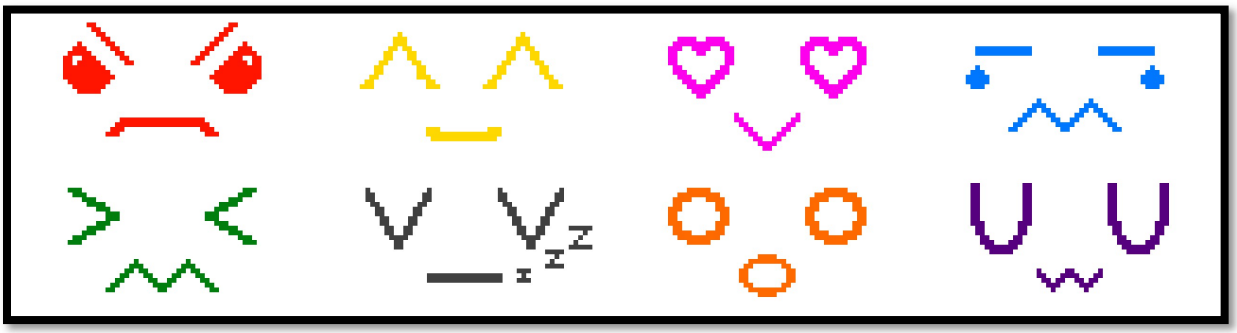


Abbildung 17 - Gesichter der LED-Matrix

3.5.1 Gesichter

Erstmal wurde versucht die Gesichter über andere Befehle (drawLine(), drawCircle()) der Bibliothek zu steuern, mit der Bibliothek können Striche, Kreise oder auch einzelne Pixel angesteuert werden, da das aber sehr viel Arbeit wäre, wurde in der Bibliothek nach einfacheren Varianten gesucht, und so wurde die Funktion drawBitmap() gefunden, die es ermöglicht gezeichnete Bilder direkt auf die Matrix zu übertragen.

Um die Gesichter zu zeichnen wurde Paint.net verwendet, das ist ein frei verfügbares Programm, das leicht zu bedienen ist, da nur einfache Gesichter zu zeichnen waren, war diese Software eine geeignete Wahl. Die Farben der Bilder sind irrelevant, da beim Senden der Bilddaten vom ESP32 zur LED-Matrix manuell eine Farbe angegeben wird. Die Farbe wird mit der Funktion matrix.color444(int r, int g, int b) ausgewählt, der erste Ausdruck in der Klammer steht für die Farbe Rot, der zweite für Grün und der dritte für Blau. Die Werte können Zahlen von 0 bis 15 annehmen, wobei 0 keine Farbe der jeweiligen Farbe bedeutet und 15 das hellster der jeweiligen Farbe bedeutet.

3.5.2 ESP32 LED-Matrix

Es wurde die Arduino IDE verwendet mit der Bibliothek ESP32-HUB75-MatrixPanel-DMA, ist es ein Einfaches die Gesichter auf dem Display dazustellen. Die LED-Matrix funktioniert einwandfrei mit dem ESP32, sie braucht 13 digitale Pins, sechs davon sind Data Bits und 7 sind Control Bits. Mit dem ESP32 sind mehr als 4096 Farben verfügbar, da der ESP32 Prozessor schneller ist als der des Arduinos, der Arduino hat 16 MHz mit diesem sind bei 40% Auslastung des Prozessors 4096 Farben also 12-bit Farben möglich mehr würde den Arduino nur unnötig belasten. Der ESP32 Prozessor hat 240 MHz mit diesem sind mehr Farben möglich doch für dieses Projekt ist kein höheres Farbspektrum nötig.

Über ein 16 Pin Kabel wird der ESP32 mit der LED-Matrix verbunden, diese braucht eine Spannung von 5V und einen Strom von 0,05-0,15 Ampere, das kommt auf eine Leistung von 0,25-0,75W, durchdiese und den Verbrauch der anderen Bauteile wurde der Akku richtig gewählt so, dass die LED-Matrix 4 Stunden laufen kann. Die Pins finden sie in Abbildung 5.

3.5.2.1 Bitmap

```
const unsigned char UwU[] PROGMEM = {
  // 'UwU face', 64x32px
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x18, 0x03, 0x00, 0x00, 0xc0, 0x18, 0x00, 0x00, 0x18, 0x03, 0x00, 0x00, 0xc0, 0x18, 0x00,
  0x00, 0x18, 0x03, 0x00, 0x00, 0xc0, 0x18, 0x00, 0x00, 0x18, 0x03, 0x00, 0x00, 0xc0, 0x18, 0x00,
  0x00, 0x18, 0x03, 0x00, 0x00, 0xc0, 0x18, 0x00, 0x00, 0x18, 0x03, 0x00, 0x00, 0xc0, 0x18, 0x00,
  0x00, 0x18, 0x03, 0x00, 0x00, 0xc0, 0x18, 0x00, 0x00, 0x18, 0x03, 0x00, 0x00, 0xc0, 0x18, 0x00,
  0x00, 0x1c, 0x07, 0x00, 0x00, 0xe0, 0x38, 0x00, 0x00, 0x0c, 0x06, 0x00, 0x00, 0x60, 0x30, 0x00,
  0x00, 0x0c, 0x06, 0x00, 0x00, 0x60, 0x30, 0x00, 0x00, 0x06, 0x0c, 0x00, 0x00, 0x30, 0x60, 0x00,
  0x00, 0x07, 0x1c, 0x00, 0x00, 0x38, 0xe0, 0x00, 0x00, 0x03, 0xf8, 0x00, 0x00, 0x1f, 0xc0, 0x00,
  0x00, 0x01, 0xf0, 0x00, 0x00, 0x0f, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x41, 0x82, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x63, 0xc6, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x36, 0x6c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1c, 0x38, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x08, 0x10, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};
```

Abbildung 18 - Bitmap

Jedes Gesicht wird in eine Bitmap umgewandelt, die dann so wie oben gezeigt als ein Char Array gespeichert wird. PROGMEM ist ein Variablenmodifikator, der dafür sorgt das die Daten statt im SRAM in den Flash Speicher gespeichert werden.

3.5.2.2 Auswahl der Gesichter

```
void drawFace(int id) {
  dma_display->clearScreen();

  switch (id) {  // draw face matching id
    case 1:      // UwU
      dma_display->drawBitmap(0, 0, UwU, 64, 32, dma_display->color444(15, 0, 15));
      Serial.println("UwU");
      break;

    case 2:  // happy
      dma_display->drawBitmap(0, 0, happy, 64, 32, dma_display->color444(15, 8, 0));
      Serial.println("happy");
      break;

    case 3:  // angry
      dma_display->drawBitmap(0, 0, angry, 64, 32, dma_display->color444(15, 0, 0));
      Serial.println("angry");
      break;

    case 4:  // love
      dma_display->drawBitmap(0, 0, love, 64, 32, dma_display->color444(15, 0, 1));
      Serial.println("love");
      break;
```

Abbildung 19 - Funktion zur Anzeige der Gesichter

Der ESP32 kriegt von der App eine ID zwischen 1 und 8 zugesendet, mit dieser wird durch eine switch-case Funktion ausgewählt welches Gesicht abgebildet werden soll. Mit der drawBitmap Funktion aus der schon genannten Bibliothek wird das Gesicht abgebildet. Die Funktion bekommt mehrere Argumente, die ersten zwei sind der Anfangspunkt der Übertragung, da die Übertragung von links nach rechts und von oben nach unten geht und die Bitmap das gesamte Display ausfüllt, wurden diese beide auf null gesetzt. Das dritte Argument ist die Bitmap, die übertragen werden soll. Das vierte ist die Breite der Bitmap und das fünfte die Höhe der Bitmap. Das letzte gibt die Farbe, die das Bild haben soll an und wird mittels der color444 Funktion bestimmt.

3.5.2.3 Bibliotheken

```
#include <Adafruit_GFX.h>     // Core Graphics Library
#include <ESP32-HUB75-MatrixPanel-I2S-DMA.h>
#include <ESP32-VirtualMatrixPanel-I2S-DMA.h>
```

Abbildung 20 – inkludierte Bibliotheken

Adafruit_GFX: https://github.com/adafruit/Adafruit-GFX-Library

ESP32-HUB75-MatrixPanel-DMA: https://github.com/mrfaptastic/ESP32-HUB75-MatrixPanel-DMA

3.6 Leiterplatte

Schaltplan und Leiterplatte wurden mittels Eagle realisiert.

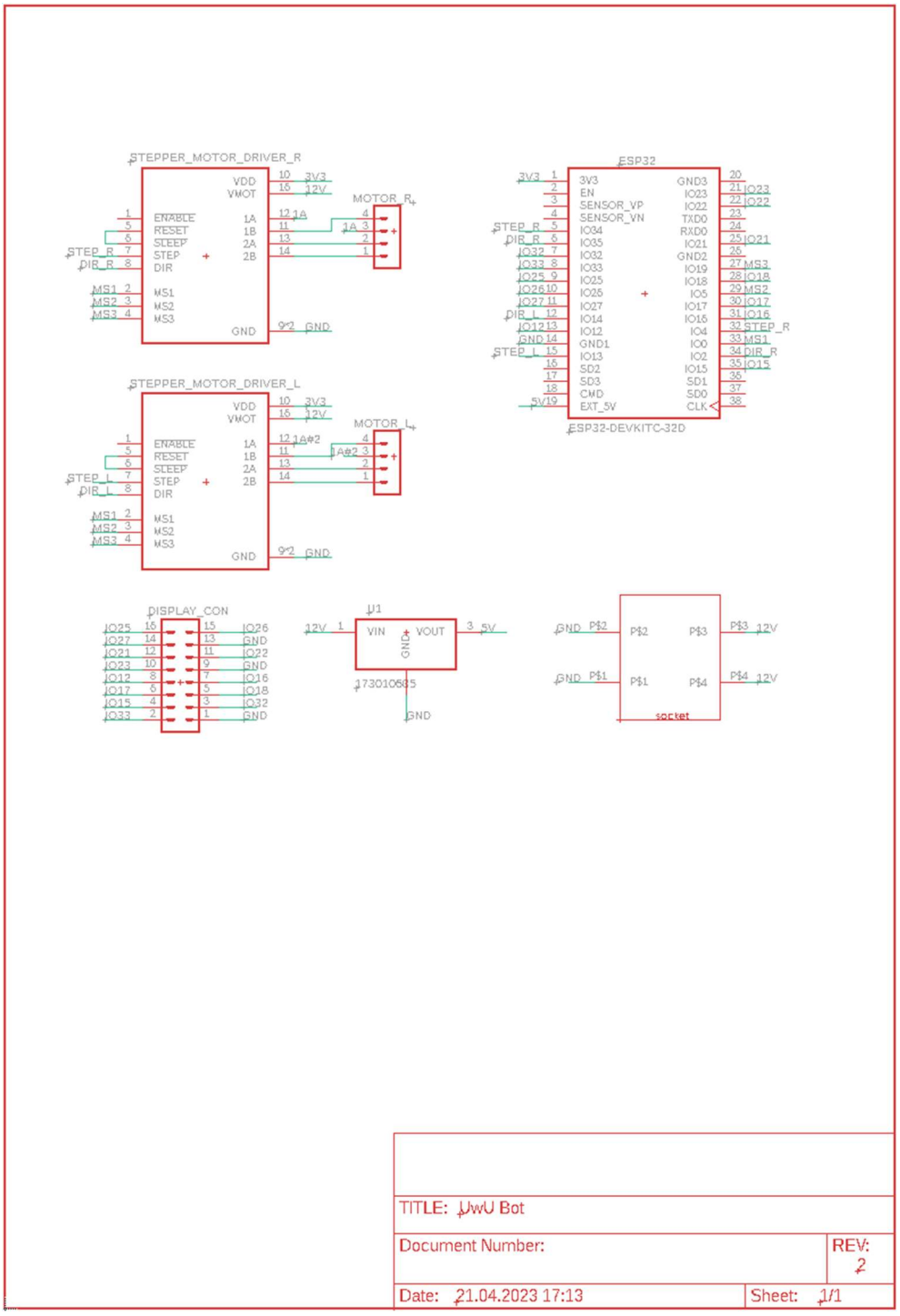


Abbildung 21 - Schaltplan

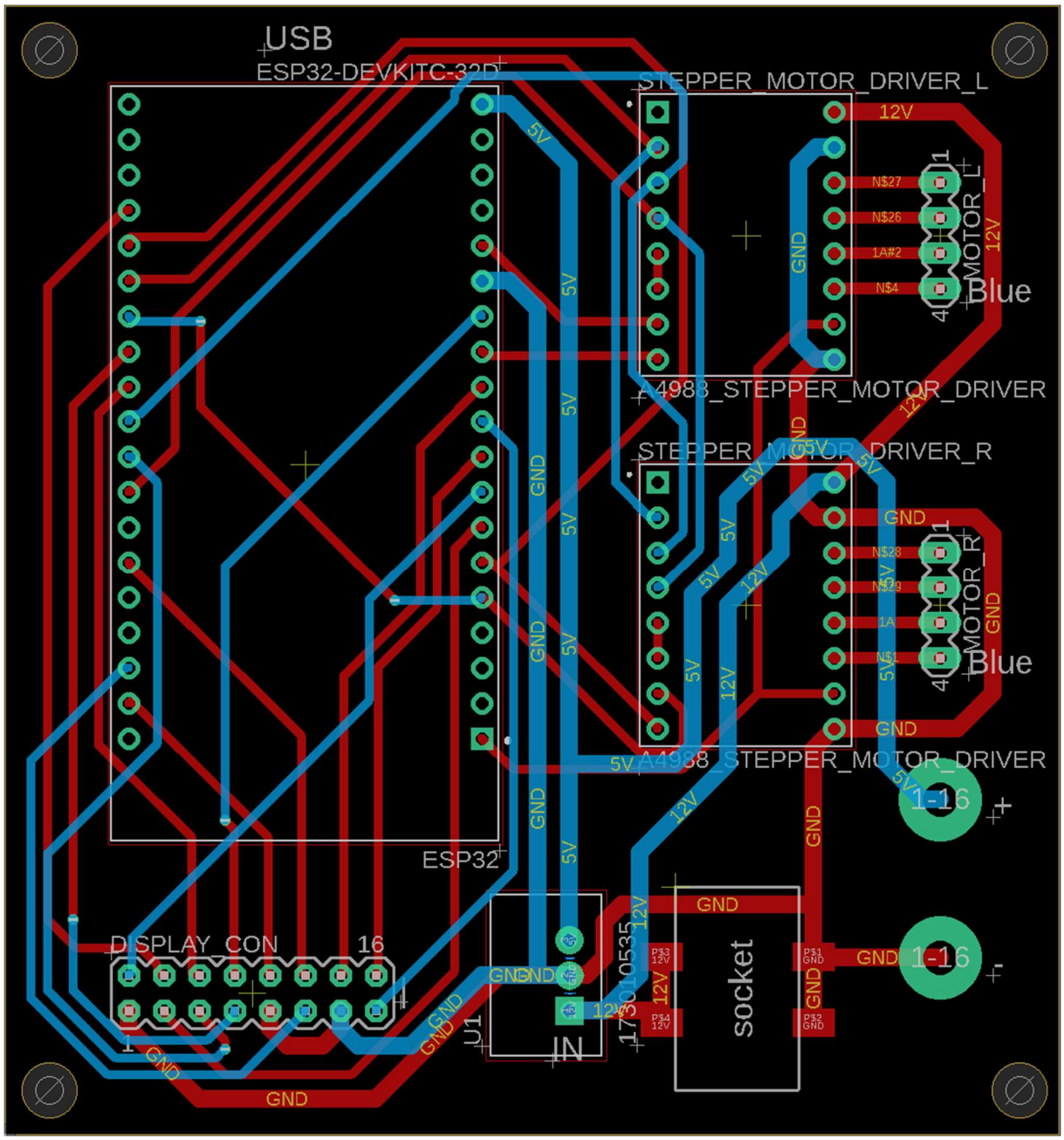


Abbildung 22 - Leiterplatte

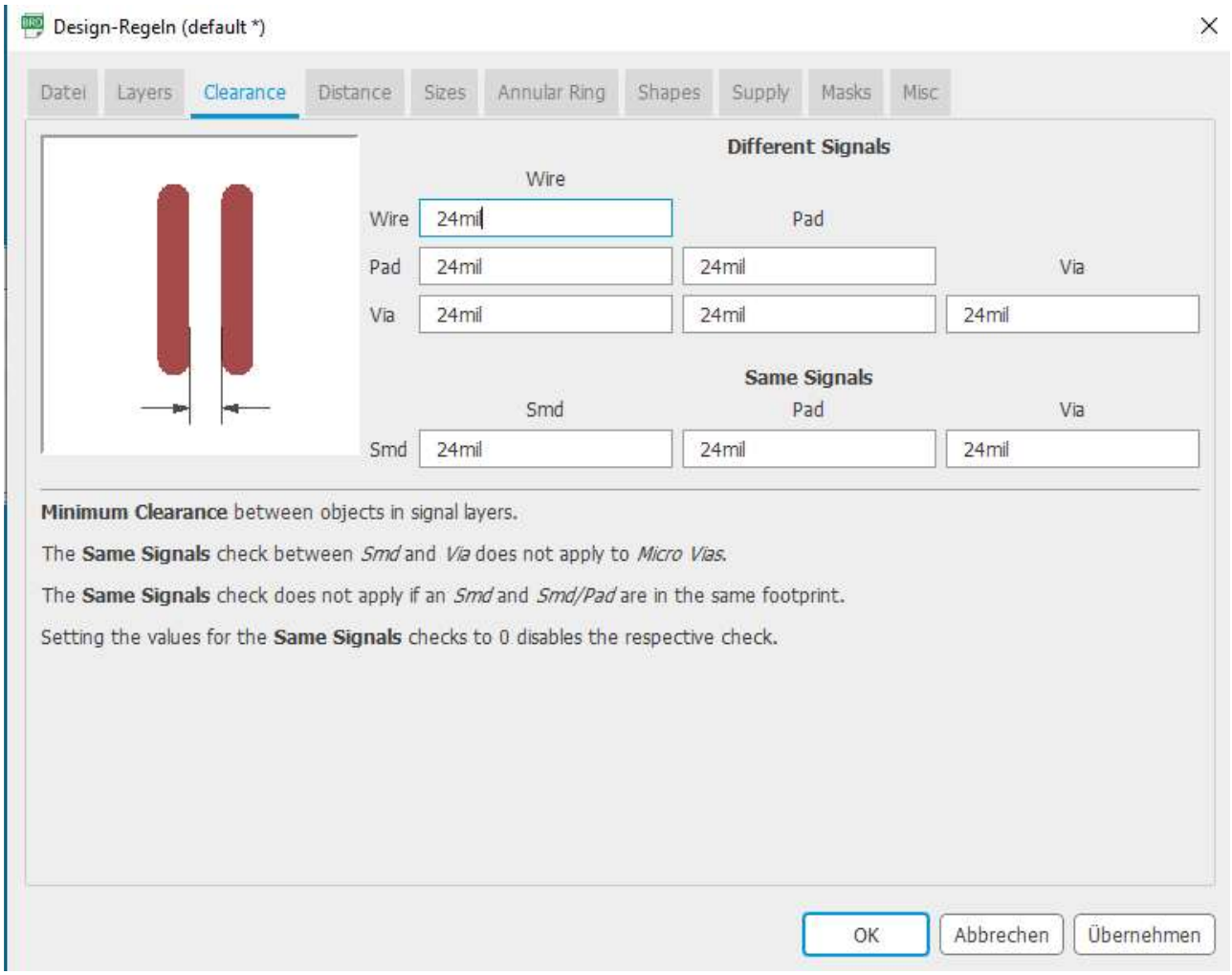


Abbildung 23 - Design-Regeln

Es wurden für alle Abstände 24 mil gewählt so sind die Leitung und Pins, nie zu nahe aneinander und es wird da druch die Fehlerqote stark verringert. Mit 24 mil sind auch alle Leitung die, die Logik übertragen gewählt worden. Die Leitungen an denen viel Strom fließt wurden mit 50 mil gewählt, es könnte weniger gewählt werden, doch mit 50 mil wird die Gefahr verringert das die Platine zu heiß werden sollte.

Die Platine hat an jeder Ecke ein Loch das 4 Millimeter breit ist, damit wird die Platine im inneren des Roboter befäßtigt. Die Vias 1-16 sind für den Plus und Minus Pol der LED Matrix dort wird die Versorgungskabel der LED-Matrix mit einer Schraube befäßtigt.

3.6.1 LED Matrix Anschlüsse

In der folgenden Abbildung werden die Inputs, Outputs und Versorgung der LED-Matrix dargestellt. Es werden nur der Data-Input und die Versorgung verwendet. Mit Hilfe der Bibliothek ESP32-HUB75-MatrixPanel-DMA, werden die Gesichter auf die LED-Matrix übertragen. Die LED-Matrix braucht eine Versorgungsspannung von 5V.

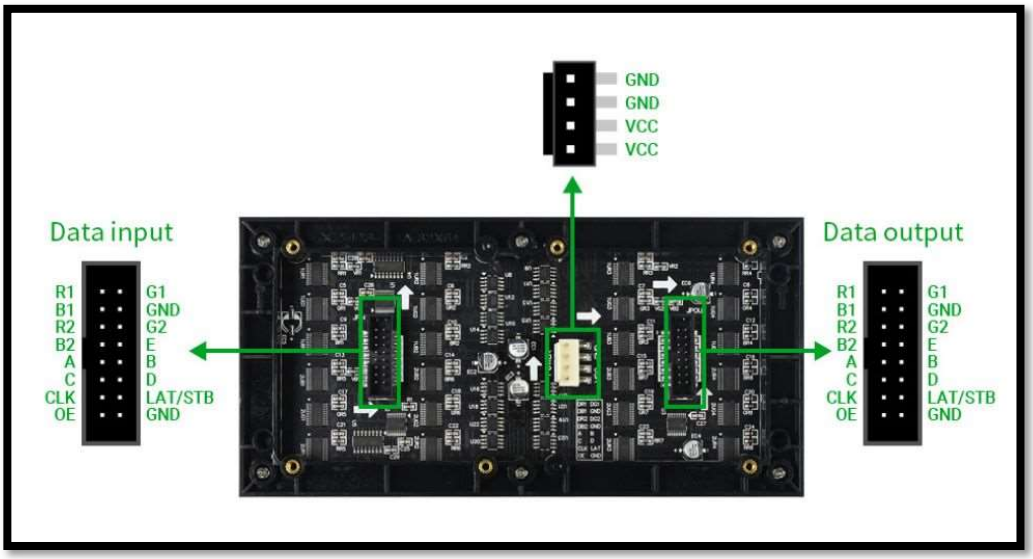


Abbildung 24 - Inputs, Outputs und Versorgung der LED-Matrix

Ein Problem wurde entdeckt, als die LED-Matrix nicht richtig funktioniert hat, dieses entstand dadurch, dass man ohne die Bibliothek andere Pins verwenden muss als mit der Bibliothek. Ohne die Bibliothek ist die Reihenfolge aller Pins vertauscht.

In der darauffolgenden Tabelle ist die Pin-Belegung der LED-Matrix am ESP32 abgebildet, ohne die besagte Bibliothek wäre es sehr viel schwieriger die Abbildung der Gesichter zu realisieren, die Bibliothek wurde auf dem ESP32 verwendet.

Tabelle 2 - LED-Matrix zu ESP32

LED Matrix zu ESP32			
Matrix	**ESP32**	**Matrix**	**ESP32**
R1	IO25	G1	IO26
B1	IO27	GND	GND
R2	IO21	G2	IO22
B2	IO23	GND	GND
A	IO12	B	IO16
C	IO17	D	IO18
CLK	IO15	LAT	IO32
OE	IO33	GND	GND

3.6.2 Schaltregler

Tabelle 3 - Pinout des Schaltreglers

Pinout	
Pin	Function
1	+Vin
2	GND
3	+Vout

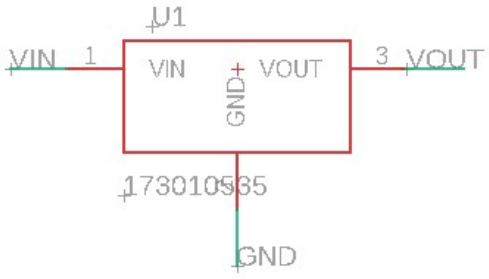


Abbildung 25 - Bauteil in Eagle

Der Schaltregler regelt die 12V des Akkus auf 5V, die 5V sind mit dem ESP32 verbunden, der ESP32 regelt diese Spannung auch noch auf 3,3V runter. Die LED-Matrix braucht 5V, während die Schrittmotor-Treiber 3,3V für die Logik brauchen. Dieses Bauteil deckt alle Spannung, die das Projekt braucht ab.

3.6.3 Buchse

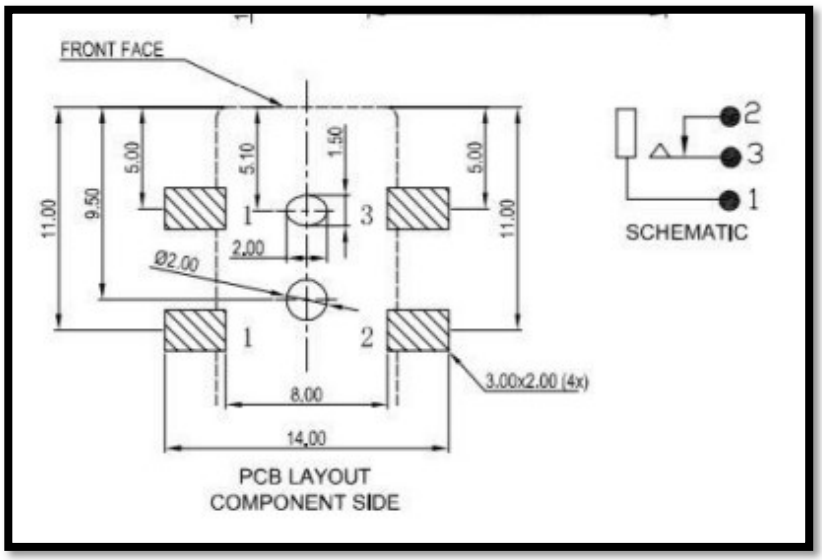


Abbildung 26 - Layout und Pinout der Buchse

Die Buchse ist der weibliche Teil des Anschlusses von dem Akku, sie bringt die Versorgung des Akkus auf die Platine.

- Pin1: Ist der Ground.
- Pin2 und 3: wurde in hier kurzgeschlossen und als Versorgung verwendet.

Mit Pin2 und 3 könnte man sicherstellen das der Stecker wirklich in der Buchse steckt, was in diesem Fall nicht von Nöten ist, da ohne diese Versorgung das ganze Gerät nicht funktioniert und so sofort festgestellt werden kann, ob der Stecker richtig steckt oder nicht.

Die Buchse war das einzige Bauteil, für das es keine vorgefertigte Bibliothek in Eagle gab, diese musste eigenhändig gefertigt werden, was ein Arbeitsaufwand von 1-2 Stunden beansprucht hat.

3.7 Gehäuse

Das Ziel des Gehäuses ist, dass es eine passende Größe und Gewicht hat, sodass das Fahrwerk nicht zu sehr überlastet ist und angenehmer fährt. Ein weiteres Ziel war, dass das Innenleben dabei auch Platz im Gehäuse hat. Hierbei wurde simpel mit einem 3D-Modell gearbeitet, wobei man sich die Größenverhältnisse angenehmer zusammensuchen konnte und kurze Vorstellungen zum fertigen Produkt erzeugen konnte.

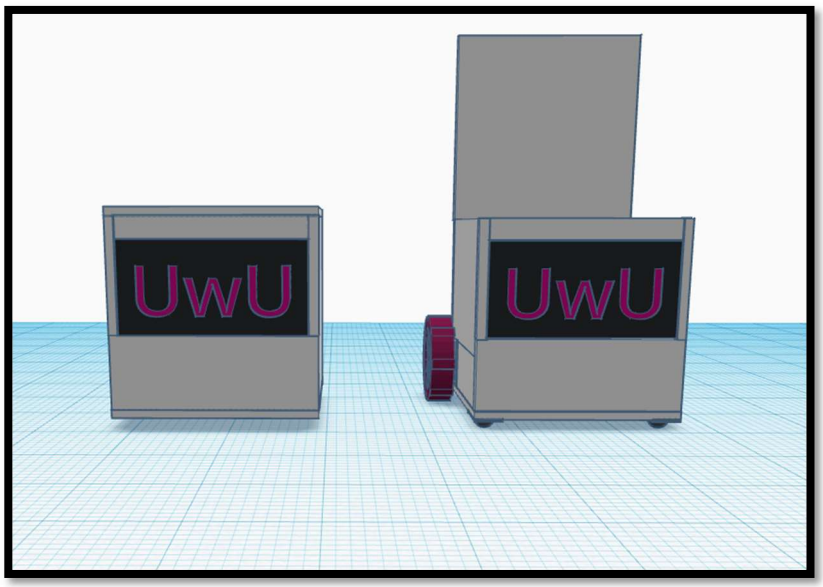


Abbildung 27 - 3D-Modell UwU-Bot frontale Ansicht

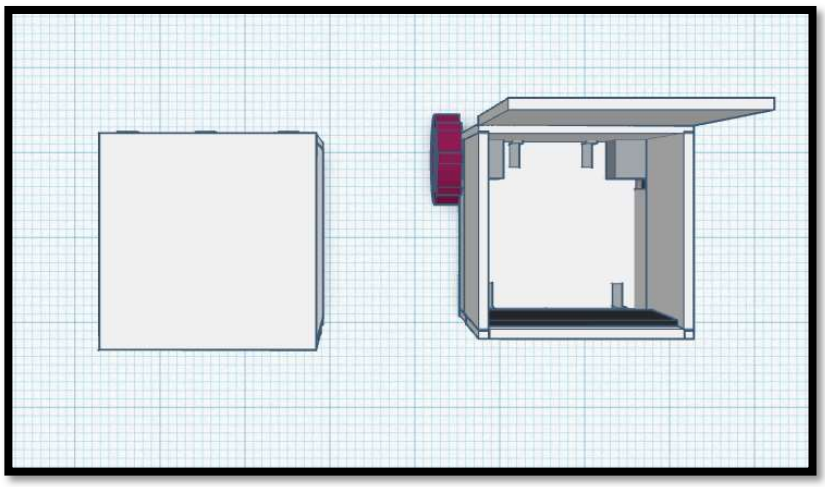


Abbildung 28 - 3D-Modell UwU-Bot Ansicht von oben

3.7.1 Material

Das Hauptziel war es den UwU-Bot mit Kunststoffplatten zusammenzubauen, jedoch wurde diese Idee schnell zunichte gemacht aufgrund schlechter Erfahrungen mit dem Material bei einem früheren Projekt. Aus diesem Grund hatten wir uns für das simpelste und auch stabileres Material entschieden, und zwar Holz, um genauer zu sein Buchenholz. Die Dicke des Materials, ließ sich mit Probematerial bestimmen. Hierbei haben wir 4 mm dicke Holzplatten gefunden und schnell bemerkt, dass dies zu dünn ist und das Doppelte um weiten besser wäre und ziemlich genau unserer Vorstellungen entspricht. Aus diesem Grund haben alle Platten eine Dicke von 8 Millimeter. Das Material wurde hierbei im nächsten Baumarkt eingekauft und dort auch zu folgenden Größen ausgeschnitten.

3.7.2 Maße

Der UwU-Bot hierbei benötigt 6 solcher Buchenholzplatten mit diversen Größen, wobei die Dicke auch mit einberechnet wurde. Der Boden und der Deckel hierbei haben eine Größe von jeweils 21 cm x 21 cm. Die Vorderseite und Rückseite wären 21 cm x 19,4 cm und die beiden Platten an den Seiten wären jeweils 19,4 cm x 19,4 cm groß. Leider gab es bei dem Zuschnitt einen kleinen Fehler vom Mitarbeiter, von einem Millimeter, was die hintere Holzplatte angeht, welcher jedoch keinen Einfluss auf die Stabilität oder Funktion hat, sondern hauptsächlich nur optisch ein wenig stört.

3.7.3 Benötigte Bauteile

Damit wir aus den Holzplatten schließlich eine Box zusammenbauen konnten, benötigten wir folgende Bauteile: 7-mal 8 M4 x 16 mm Schrauben mit dazugehörigen Muttern, 12-mal Winkelverbinder, 3-mal 1,7 cm x 1,4 cm Messingscharniere, 1-mal Lochband und 1-mal Leim.

3.7.4 Wichtige Anmerkungen

Natürlich wurde auch bedacht, dass das Fahrwerk überhaupt so eine Leistung mitbringt und das Gewicht hierbei kein Problem ist. Wie bereits erwähnt soll dieser dann auch auf zwei Möbelrollen gestützt sein, die ebenfalls kein Problem mit dem Gewicht haben. Dazu wurde auch bedacht, dass der UwU-Bot diverse Ausschnitte braucht für die LED-Matrix und das Fahrwerk.

3.7.5 Zusammenbau

Für die Zusammensetzung des Gehäuses wurde auch ein entsprechender Arbeitsplatz benötigt. Als passender Ort kam uns hierbei die KUV-Werkstatt in den Kopf, weshalb der Zusammenbau auch dort stattfand.

3.7.5.1 Lasercutter

Zuerst mussten wir den Bereich wo die LED-Matrix zum Vorschein kommt mit einen Lasercutter schneiden, da die Maße dabei millimetergenau sind. Dafür haben wir den verantwortlichen Fachlehrer für die Lasercutter aufgesucht und gefragt ob es möglich wäre die Fläche hierbei aus der Holzplatte herauszuschneiden. Leider kam es da dann auch zu einem kleinen Schnittfehler aufgrund einer Fehlmessung des Fachlehrers, jedoch machte dies wiederum keine Probleme, was die Funktion angeht, sondern war hauptsächlich nur ein visueller Aspekt der leicht störte. Der Schnitt sollte hierbei 9,6 cm x 19,2 cm und war dann jedoch ein Ausschnitt von 10 cm x 19,2 cm.

3.7.5.2 Maschinen

Vorerst begann alles mit der Vorbereitung der einzelnen Maschinen. Dazu gehörte der Bohrer, wo ein 4 mm breiten Bohrer-Aufsatz zum Einsatz kam und einer Holzbandsäge zum Ausschneiden der Bereiche für das Fahrwerk.

3.7.5.3 Bohren

Insgesamt kam es zu 52 verschiedenen Bohrungen, darunter 4 für das Lochband und der Rest für die Befestigung. Dabei mussten in der unteren Holzplatte 18 Bohrungen stattfinden, in der vorderen Holzplatte 8, in der hinteren Holzplatte 10, bei der rechten Holzplatte wiederum 8 und bei der linken auch 8.

3.7.5.4 Sägen

Wie bereits erwähnt werden auch zwei Ausschnitte an beiden Seiten für das Fahrwerk benötigt, die dann selbst ausgeschnitten wurden. Die Maße dafür betragen jeweils 4,4 cm x 5,5 cm.

3.7.5.5 Einsatz von Leim

Aufgrund gewollter besserer Stabilität wurde zusätzlich zu den Schrauben auch Leim zwischen den Holzplatten eingesetzt, damit diese auch möglichst robust werden.

3.7.5.6 Feinschliff

Aufgrund des Höhenunterschieds der Möbelrollen und des Fahrwerks, den wir durch das 3D-Modell auch schnell bemerken konnten, kamen wir zum Entschluss, das ausgeschnittene Holz wiederzuverwenden und somit eine Erhöhung zu erstellen.

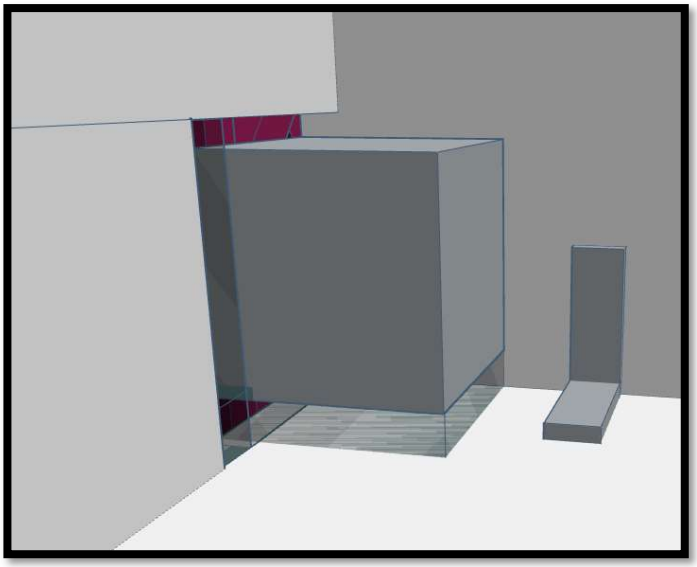


Abbildung 29 - Erhöhung des Fahrwerks

3.7.6 Weitere Bemerkungen

Zu dem Gehäusebau gehört auch die Ordnung der einzelnen Bauteile im Gehäuse selbst. Dazu gehört auch die notwendige Befestigung des Fahrwerks. Dafür wird eine Art Polsterung verwendet damit die Vibrationen der Motoren keine zu lauten unangenehmen Geräusche erzeugen. Dafür haben wir das Füllmaterial der Motoren selbst die bei der Lieferung dabei waren verwendet und so für die Lärmdämmerung gesorgt. Damit jedoch das Fahrwerk dann auch stabil im Gehäuse befestigt wird kam das Lochband zum Einsatz. Wie bereits angemerkt wurden 4 extra Löcher dafür gebohrt. Die dienen dazu, dass die Schrauben durchkommen und so mit den Muttern das Lochband dann auch an den jeweiligen Platten befestigt und das Fahrwerk so stabil wie möglich quasi montiert. Weiteres die LED-Matrix. Diese passt wie angegossen in den Zuschnitt, der durch den Lasercutter erzeugt wurde. Jedoch nur in der Breite, was dennoch mehr als genug reicht, dass die LED-Matrix perfekt in dieser Öffnung stecken bleibt und keine Probleme weiters mit sich bringt.

3.7.7 Design

Nachdem nun der Zusammenbau ein Ende fand, kommt es nun zu den optischen Aspekten des Gehäuses. Dazu gehört das Lackieren. Dies erfolgt mit dem Schleifen der einzelnen Platten aufgrund der rauen Oberflächen. Das Lackieren erfolgt dann mit einem Holzlackspray aus dem handelsüblichen Baumarkt und wird dabei von Anfang bis Ende sozusagen lackiert.

Abbildung 30 - Vorher-Nachher-Bild

3.7.8 Fertiges Design

Abbildung 31 - Gesichtsausdrücke beim fertigen Bot

3.8 Reflexion der Projektrealisierung

3.8.1 Vorschläge für weitere Entwicklungen/Verbesserungen

Man könnte dem Roboter einen Lautsprecher einbauen, mit dem dieser in die App eingegebene Texte mithilfe von Text-To-Speech sagen könnte oder immer, wenn eine Emotions-Änderung stattfindet, ein bestimmtes Geräusch von sich gibt. Es könnte auch eine Sprachsteuerung in die App eingebaut werden, so dass man einfach ins Mikrofon des Handys sprechen kann, um den Roboter zu steuern.

Man könnte den UwU-Bot auch personalisieren, in dem man ihm durch Klettverschlüsse zum Beispiel Ohren, Hüte oder Arme geben kann und so den Roboter passend für sich personalisiert. Außerdem wäre eine Verbesserung, die wir unbedingt uns vorgenommen hatten, statt normalem und regulären Buchen- oder Eichenholz, Zirbe aufgrund des Geruchs zu verwenden, jedoch wäre das preislich viel zu teuer ausgefallen, weshalb wir dies nicht taten. LED-Streifen unter dem UwU-Bot, die als Unterbodenbeleuchtung wie bei Autos dienen und den dadurch den UwU-Bot noch um einiges cooler machen, wären auch eine großartige Idee.

Ein spannendes Feature wäre noch, dass man den Roboter selbst programmieren könnte, indem man eine leicht zu bedienende Benutzeroberfläche macht, ähnlich wie Scratch. Es wäre auch eine Option, die Höchstgeschwindigkeit des Roboters in der App festlegen zu können, anstatt diese im Code fest zu verankern. Man könnte auch eigene Gesichter auf der LED-Matrix anzeigen lassen, die man über die App an den Roboter schicken kann. Ein weiterer großer Schritt wäre es Animationen zu ermöglichen, die man möglicherweise auch einfach in der App erstellen kann.

Die Möglichkeit eine Uhr anzeigen zu lassen wäre ein nützliches Feature, wenn der Roboter nur im Zimmer steht, so dass er auch im Alltag einen nutzen hat. In Verbindung mit den oben angesprochenen Lautsprechern könnte man dann auch logischerweise eine Art Wecker/Timer Funktion hinzufügen. Es wäre auch praktisch, wenn man den Roboter, wenn er sich im Wecker-Modus befindet, an ein Ladegerät anstecken könnte, so dass dieser nicht leer wird, bevor der Wecker läutet.

Ich versichere,

dass ich die vorliegende Diplomarbeit selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe. Inhalte, die wörtlich oder sinngemäß aus Veröffentlichungen oder anderen Quellen entnommen wurden, sind als solche kenntlich gemacht. Die Arbeit ist in gleicher oder ähnlicher Form noch nicht veröffentlicht und noch keiner Prüfungsbehörde vorgelegt worden.

Graz, am ....

Graz, am ....

Graz, am ....